Abstract

The feature set used with a classifier can have a large impact on classification performance. This paper presents a set of shrinkage-based features for Maximum Entropy and other classifiers in the exponential family. These features are inspired by the exponential class-based language model, Model M. We motivate the use of these features for the task of text classification and evaluate them on a natural language call routing task. The proposed features along with a new word clustering method result in significant improvements in action classification accuracy over typical word-based features, particularly for small amounts of training data.

Index Terms: shrinkage-based features, call routing, model M, exponential models

1. Introduction

Spoken dialogue systems (SDSs) provide natural access to information and services over the phone. SDSs in general and natural language call routing in particular are arguably among the most successful speech applications, as this technology is used in almost all large contact centers [1, 2]. The call routing task entails directing a customer’s call to an appropriate destination within a call center on the basis of customer interaction. Typically, call-routing systems require two statistical models. The first model performs speech recognition and transcribes what the caller says. The second is the Action Classification (AC) model that takes the transcribed utterance and maps it to an action that hopefully fulfills the caller’s request.

Building a robust and highly accurate call-routing system requires large amounts of manually transcribed and labeled data from the domain of the application [1, 3]. The human labeler assigns one or more of the predefined action classes, or call types, to each utterance. This requires full understanding of the domain of application. Naturally, it is expensive to manually label large amounts of data. In the early development phases of such systems, little annotated data is typically available. Therefore, superior machine learning techniques can go a long way toward achieving adequate action classification performance.

In previous work, shrinkage-based exponential language models [4, 5, 6], particularly Model M, were shown to achieve some of the largest gains over word n-gram models reported in the literature on a variety of domains. These gains were accomplished using just lexical information, namely by adding class n-gram features, and without using additional syntactic and semantic information. Inspired by these results, we propose to use similar features for the action classification task.

We note that the problem of call routing is similar to that of text categorization. In text categorization, one attempts to identify which of a given set of topics matches a document. While the amount of text available for making a classification decision is typically larger, the same set of techniques are used both in text categorization and call routing. Therefore, we believe the proposed features should be applicable to text categorization tasks as well.

This paper is organized as follows: Section 2 describes the Maximum Entropy classifier and Section 3 describes the proposed shrinkage-based features. Section 4 describes the algorithms used to induce word classes. Section 5 presents experimental results followed by conclusions in Section 6.

2. Maximum Entropy Classifiers

The Maximum Entropy (MaxEnt) method is a statistical modeling framework used in many natural language processing tasks [13]. In this framework, one can combine multiple overlapping information sources in an effective manner. Specifically, we take the probability $P(C|W)$ of a particular action class $C$ given the caller’s spoken word sequence $W$ to be

$$P(C|W) = \frac{\exp(\sum_{f} \lambda(f(C,W))}{\sum_{C'} \exp(\sum_{f} \lambda(f(C',W)))}$$

(1)

The $f_i$ are indicator functions, or features, which are “activated” based on computable features of the word sequence; e.g., a feature may be active if a particular word or word pair appears, or if the parse tree contains a particular tag, etc.1 The denominator is a normalization factor that forces probabilities to sum to 1. As is common practice, we include unigram (single word) and bigram features in our baseline model, one for each such n-gram that occurs in the training data. Our MaxEnt models are trained using the improved iterative scaling algorithm [13] with Gaussian prior smoothing [12] using a single universal variance parameter of 2.

3. Shrinkage-based Features

Before describing shrinkage-based features for call routing, it is essential to introduce Model M and to briefly summarize the background work that led to Model M.

3.1. Model M

Model M is a class-based exponential n-gram model that has been shown to significantly outperform word-based n-gram models on a variety of domains, tasks and languages. Shrinkage-based exponential language models in general and Model M in particular were motivated as ways to shrink a word

1Generally, we use the term feature to refer to a computable feature of $W$. In fact, we take each $f_i$ to be active for only a single $C$. In general, there will be many $f_i$ for each feature of $W$, one for each $C$ that feature co-occurs with in the training data.

Shrinkage-Based Features for Natural Language Call Routing

Ruhi Sarikaya, Stanley F. Chen, Bhuvana Ramabhadran

IBM T.J. Watson Research Center, Yorktown Heights, New York, 10588
{sarikaya,stanchen,bhuvana}@us.ibm.com
n-gram model. That is, when training and test data are drawn from the same distribution, it has been empirically found for many types of exponential language models covering a wide range of conditions that

\[ \log P_{\text{test}} \approx \log P_{\text{train}} + \frac{\gamma}{D} \sum_i |\lambda_i| \]  

(2)

where \( P_{\text{test}} \) and \( P_{\text{train}} \) denote test and training set perplexity; \( D \) is the number of words in the training data; \( \lambda_i \) are regularized (i.e., smoothed) estimates of the model parameters; and \( \gamma \) is a constant independent of domain, training set size, and model type [4]. Thus, one can improve test performance if one can shrink \( \sum_i |\lambda_i| \) (or model size) while maintaining training set performance.

Model M exploits the above relationship by clustering the vocabulary and using word classes to shrink model size. If each word \( w \) is mapped to a single class, we can write

\[ p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p_{\theta}(c_j | c_{j-2} \cdots c_{j-1}, w_{j-2}w_{j-1}) \times \prod_{j=1}^{l} p_{\theta}(w_j | w_{j-2}w_{j-1}c_j) \]

where \( p_{\theta}(y|\theta) \) denotes an exponential \( n \)-gram model, and where \( p_{\theta}(y|\theta_1, \theta_2) \) denotes a model containing all features in \( p_{\theta_1}(y|\theta_1) \) and \( p_{\theta_2}(y|\theta_2) \).

Even though the above formulation assumes each word belongs to a single class (i.e., hard classing), it can also be extended to soft classing where each word can have multiple class memberships, as we have done in [8].

3.2. Model M features

Given the remarkable success of Model M for language modeling, we wish to see whether these ideas can be extended to call routing and text classification. In other words, we would like to discover whether features used by Model M for language modeling are also helpful for classification problems, despite the differences between these domains. Notably, in language modeling one predicts the next word from among a vocabulary of up to several hundred thousand words, while in call routing the number of call types is on the order of a few dozen. In language modeling one conditions on the past few words, while in call routing we condition on the entire utterance. Table 1 has the unigram (1gr) and bigram (2gr) feature sets we propose to use in the experiments. Note that the bigram features listed in the table contain the unigram features as a subset. The bigram language model of the form:

\[ p(w_j | w_{j-1}) \equiv p(c_j | c_{j-1}) p(w_j | c_j) \]  

(3)

Table 1: Different sets of shrinkage-based features. Note that 2gr features include the 1gr features as a subset

be generated by using domain knowledge particular to spoken dialog applications [8]. In this work, we use the same clustering algorithms, inducing word classes from the utterances to be classified (and ignoring the call type labels). We speculate that which classes work best may be somewhat independent of the target classification task, though this belief should be further investigated.

4. Word Clustering Methods

4.1. IBM word clustering method

The automatic word classing algorithm used in the original Model M work partitions the vocabulary into a specified number of classes in an attempt to maximize the bigram mutual information between word classes [10]. The algorithm starts by assigning each of the most frequent words to their own class and the remaining words to the final class. Then, the exchange algorithm is performed: Individual words are moved to another class if this improves the class bigram mutual information, until no more such moves are possible.

4.2. Enhanced word clustering method

While IBM clustering is the most popular clustering algorithm for language modeling, it was developed independently of Model M. Recently, a new clustering method tailored specifically to Model M has been shown to give better performance with this model [7]. Optimizing the IBM classing objective function is equivalent to optimizing the training set likelihood of a class bigram language model of the form:

\[ p(w_l | w_{l-1}) \equiv p(c_l | c_{l-1}) p(w_l | c_l) \]

The new algorithm modifies this framework in several ways. First, unlike in the above equation where the current word and class are predicted conditioning only on the previous class, previous words are also conditioned on. Secondly, instead of optimizing training set likelihood, estimated test set likelihood is optimized, for otherwise conditioning on previous classes would not help above conditioning directly on previous words. Finally, trigram rather than bigram information is used. The new classing method also uses a different search algorithm to find the clustering that optimizes the objective function. In addition to the exchange algorithm, class merging and class splitting moves are also greedily applied if they improve the objective function. In speech recognition experiments with Wall Street Journal training sets of up to 23M words, the new clustering algorithm outperforms IBM classing with Model M by up to 1% absolute in word-error rate [7].

5. Experimental Results

We consider two tasks: Task 1 uses data from a technical assistance hotline for package shipment for a Fortune 500 company [11]. The natural language call routing system selects
from one of 35 call types. The training data has 27K utterances amounting to 178K words. Training sets containing \{1K, 2K, \ldots, 9K, 10K, and 27K\} utterances are created to evaluate the proposed features over various training set sizes. A separate data set containing 5644 utterances is used as the test set and another set of 3208 utterances is used as the development data. All of these data sets are hand-labeled with call types. On average, each utterance is about 6.7 words long. The training vocabulary contains about 3.7K words.

The data for Task 2 is also from a large Fortune 500 company in the financial services domain. There are 54 call types modeled by the action classifier. The complete training data contains about 51K sentences, from which training sets of \{1K, 2K, \ldots, 9K, 10K, 15K, 25K, 51K\} utterances are created. The utterances are 4.6 words long on average. The complete training vocabulary contains about 1.6K words. Before training the classifiers, Task 2 data is preprocessed so that fund names and plan names are pre-tokenized into a generic class name, which reduces the vocabulary significantly. The development and test data contain about 2.8K sentences each. For both tasks, the development data is used to find the number of word clusters that gives the highest classification accuracy on the development data. Then, action classification models trained with these word clusters are evaluated on the test data. Task 2 is particularly interesting as the fund and plan names are already tokenized into two separate generic classes. This kind of preprocessing is reasonable in many real world applications, where available domain knowledge can be incorporated into the data and the task.

As eq. (2) shows, as training set size \(D\) increases, the last term decreases. In the limit, the last term goes to zero and there will be no benefit from shrinking model size. In language modeling, due to large vocabulary sizes and large n-gram orders, a vast amount of data is needed to reduce the last term enough so that there is no benefit from model shrinkage. That is why even when using training sets of over a billion words, we still see improvements from Model \(M\) [6, 9].

In Table 2, we present the results in the package shipment domain for the baseline lexical features (MaxEntBase) and for the Model-M-based features (MaxEntM) over a range of training set sizes. Both unigram and bigram classifiers are trained on each data set. To clarify, the unigram version of the baseline system has a single feature active for each observed word in a given utterance (i.e., a bag of words model), and the bigram version has active features for all bigrams as well as unigrams in a given utterance.

The results in Table 2 show that for unigram models, MaxEntM with IBM classing improves the classification accuracy from 0.5\% to 0.8\% absolute over the baseline (i.e., MaxEntBase) on data sets of up to 6K utterances. Above 6K, the improvements are from 0.1\% to 0.5\%. Using enhanced classing provides further improvement. For training sets with 1K to 6K utterances, the improvements over the baseline are from 0.8\% to 1.5\% absolute. For the larger training sets, the improvements are between 0.1\% and 1.0\%.

The improvements are large for smaller training data sizes (e.g., 1.5\%, 1.2\%, 1.5\% and 0.8\% for 1K, 2K, 3K and 4K, respectively) and tend to decrease as the amount of training data increases, as expected given eq. (2). For the bigram case, we observe similar trends across all feature sets. FeatSetA, FeatSetB, FeatSetC and FeatSetD all show consistent gains over the word-based baseline. FeatSetB (i.e., bigram \(Model M\) features) with IBM classing achieves 0.1\%–0.6\% improvements over the baseline across different training sizes with an average improvement of 0.3\%. With enhanced classing, the improvements go up consistently. The improvements cover a range of 0.0\% to 1.3\%, with an average of 0.6\%. The improvements are larger when the training data size is smaller. For example, when the training data size is below 6K utterances, the average improvement is 0.8\%. Comparing alternative bigram feature sets, namely, FeatSetA, FeatSetC and FeatSetD to FeatSetB and the bigram baseline, we observe overall FeatSetC is better than the other feature sets and provides further improvements over FeatSetB. The overall improvements with FeatSetC over the baseline for 1K, 2K, 3K and 4K training data sizes are 1.4\%, 0.9\%, 1.4\% and 0.7\%, respectively. We also observe that for smaller training set sizes, the unigram models may provide better accuracy than the bigram models for both MaxEntBase and MaxEntM. This is likely due to data sparsity, as more parameters have to be estimated for the bigram case and there may not be sufficient data to do so.

Table 3 presents the results for the financial transaction task. We again trained our MaxEnt action classifier using various amounts of training data. We see significantly larger gains across the board. With enhanced classing, MaxEntM achieves 2.6\% to 5.3\% absolute improvements over MaxEntBase across different training data sizes. Once again, enhanced classing provides further improvements over IBM classing, where the average improvement is 1.3\%. MaxEntM accuracies with both classings steadily improve as the training data size gets larger. However, the baseline model obtains an unexpected result when the entire 51K set is used to train MaxEntBase. The result is significantly worse than those obtained with the 15K and 25K datasets. It is not surprising to observe some occasional improvements over a model that is trained on the entire data when a subset of the data is used to train the models due to random sampling. We further looked into this by running additional experiments with several random and uniform sampling experiments to create the smaller data sizes. The results are in agreement with what is presented in the table. Our explanation is that the training data is nonhomogenous (relative to test data), with some parts being closer to the test data than others, and the good parts happen to overlap with the small data sets. Interestingly, the results with the baseline bigram features are inline with our expectations. In other words, as the data size increases the performance consistently gets better.

For the bigram case, the improvements follow roughly the same pattern as what we observed in Task 1. However, the gains are significantly larger. For example, FeatSetB with enhanced classing achieves 2.3\%, 2.6\%, 1.6\% and 2.9\% absolute improvements for the 1K, 2K, 3K and 5K training data sizes, respectively, over baseline bigram word features. Unlike the previous task where the gains shrink as the training data size increases, here we continue to observe significant gains. For example, FeatSetB achieves 1.9\%, 1.2\% and 1.2\% better performance than the baseline for the 15K, 25K and 51K data sets, respectively. Here, we observe that FeatSetC is again consistently better than FeatSetB and the other feature sets. On average, the performance numbers for FeatSetC is 0.5\% better than those for FeatSetB. FeatSetC provides a remarkable 2.4\% absolute average improvement over MaxEntBase across all training data sizes. Enhanced classing consistently provides additional improvements over baseline IBM classing across all feature sets. For Task 2, using bigram features always provides better results than using unigram features.
Table 2: PACKAGE SHIPMENT TASK: Accuracy for baseline Maximum Entropy (MaxEntBase) and Maximum Entropy M (MaxEntM) classifiers. IBM/Enhanced stands for IBM word clustering and the new Enhanced word clustering schemes, respectively.

<table>
<thead>
<tr>
<th>Data</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>1gr features</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetA</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetB</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetC</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>74.0</td>
<td>74.7</td>
<td>75.7</td>
<td>75.0</td>
<td>75.7</td>
<td>75.3/76.1</td>
<td>75.0</td>
<td>75.7</td>
<td>75.5/77.0</td>
<td>75.4/76.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2K</td>
<td>75.9</td>
<td>74.4/76.5</td>
<td>73.8</td>
<td>75.7/76.1</td>
<td>75.0/76.1</td>
<td>75.5/77.0</td>
<td>75.4/76.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3K</td>
<td>77.6</td>
<td>78.4/78.6</td>
<td>76.7</td>
<td>75.0</td>
<td>74.9/79.3</td>
<td>78.4/79.3</td>
<td>79.0/79.6</td>
<td>79.1/79.1</td>
<td>79.6/80.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5K</td>
<td>78.8</td>
<td>78.0/81.3</td>
<td>79.4</td>
<td>78.6/81.3</td>
<td>79.6/84.0</td>
<td>80.8/84.0</td>
<td>80.7/84.5</td>
<td>81.3/84.9</td>
<td>81.7/85.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10K</td>
<td>81.5</td>
<td>82.8/82.9</td>
<td>81.1</td>
<td>80.8/84.0</td>
<td>83.6/84.0</td>
<td>84.0/84.0</td>
<td>84.8/84.5</td>
<td>85.4/85.5</td>
<td>85.8/86.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25K</td>
<td>84.3</td>
<td>83.0/87.1</td>
<td>83.5</td>
<td>85.4/86.5</td>
<td>85.0/85.5</td>
<td>85.8/86.0</td>
<td>86.4/85.7</td>
<td>86.9/86.8</td>
<td>86.7/87.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51K</td>
<td>85.1</td>
<td>84.9/86.8</td>
<td>85.8</td>
<td>86.8/88.0</td>
<td>87.4/86.8</td>
<td>88.2/87.0</td>
<td>88.6/87.8</td>
<td>90.0/90.0</td>
<td>90.7/90.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: FINANCIAL TRANSACTION TASK: Accuracy for baseline Maximum Entropy (MaxEntBase) and Maximum Entropy M (MaxEntM) classifiers. IBM/Enhanced stands for IBM word clustering and the new Enhanced word clustering schemes, respectively.

<table>
<thead>
<tr>
<th>Data</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>1gr features</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetA</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetB</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetC</th>
<th>MaxEntBase</th>
<th>MaxEntM</th>
<th>2gr FeatSetD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>73.9</td>
<td>74.4/76.5</td>
<td>73.8</td>
<td>75.7/76.1</td>
<td>75.0/76.1</td>
<td>75.5/77.0</td>
<td>75.4/76.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2K</td>
<td>75.9</td>
<td>78.4/78.6</td>
<td>76.7</td>
<td>78.6/79.3</td>
<td>78.4/79.3</td>
<td>79.0/79.6</td>
<td>79.1/79.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3K</td>
<td>77.6</td>
<td>80.3/81.3</td>
<td>79.4</td>
<td>80.8/84.0</td>
<td>83.6/84.0</td>
<td>84.0/84.0</td>
<td>84.8/84.5</td>
<td>85.4/85.5</td>
<td>85.8/86.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5K</td>
<td>78.8</td>
<td>82.8/82.9</td>
<td>81.1</td>
<td>84.0/84.0</td>
<td>83.6/84.0</td>
<td>84.0/84.0</td>
<td>84.8/84.5</td>
<td>85.8/86.0</td>
<td>86.4/85.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10K</td>
<td>81.5</td>
<td>82.9/83.9</td>
<td>83.5</td>
<td>85.4/86.5</td>
<td>85.0/85.5</td>
<td>85.8/86.0</td>
<td>86.4/85.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25K</td>
<td>84.3</td>
<td>83.0/87.1</td>
<td>84.8</td>
<td>86.3/86.8</td>
<td>86.3/86.8</td>
<td>86.3/86.8</td>
<td>86.7/87.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51K</td>
<td>85.1</td>
<td>84.9/86.8</td>
<td>85.8</td>
<td>87.4/86.8</td>
<td>87.8/88.1</td>
<td>87.8/88.1</td>
<td>88.2/87.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. Conclusions

We present a set of automatically-induced features for a Maximum Entropy classifier for natural language call routing. These features are inspired by Model M, a shrinkage-based exponential language model. The fact that these features do not require additional annotation and are easy to extract from data in an unsupervised fashion makes them particularly appealing. The experimental results on two call-routing tasks show consistent gains over lexical features on small to medium training set sizes. As training data increases, the gains diminish. We also evaluated two word clustering schemes: IBM word classing and enhanced word classing. We found that the enhanced classing algorithm provides further improvement over the baseline word classing scheme.

7. References